Backpropamine: meta-training self-modifying neural networks with gradient descent

Thomas Miconi* **Aditya Rawal** **Jeff Clune** **Kenneth O. Stanley**
Uber AI Labs
{tmiconi|aditya.rawal|jeffclune|kstanley}@uber.com

Abstract

The impressive lifelong learning in animal brains is primarily enabled by plastic changes in synaptic connectivity. Importantly, these changes are not passive, but are actively controlled by neuromodulation, which is itself under the control of the brain; thus, evolution has meta-trained the brain to modify itself adequately for efficient lifelong learning. Here we show for the first time that artificial neural networks with such neuromodulated plasticity can be trained with gradient descent. Extending previous work on differentiable Hebbian plasticity, we propose a differentiable formulation for the neuromodulation of plasticity. We show that neuromodulated plasticity improves the performance of neural networks on both reinforcement learning and supervised learning tasks. In particular, a neuromodulated plastic LSTM with 4.8 million trainable parameters outperforms a standard LSTM with the same total number of parameters on a benchmark language modeling task.

1 Introduction

Neural networks that deal with temporally extended tasks must be able to store traces of past events. Often this memory of past events is maintained by neural activity reverberating through recurrent connections; other methods for handling temporal information exist, including memory networks [1] or temporal convolutions [2]. However, in nature, the primary basis for long-term learning and memory in the brain is *synaptic plasticity* – the automatic modification of synaptic weights as a function of ongoing activity [3, 4].

Importantly, these modifications are not a passive process, but are actively modulated on a moment-to-moment basis by dedicated mechanisms: the brain can "decide" where and when to modify its own connectivity, as a function of its inputs and computations. This *neuromodulation* of plasticity, involving several chemicals (particularly dopamine [5, 6, 7, 8]), plays an important role in learning and adaptation [9, 10, 11].

Small networks ($\sim$100 neurons) with neuromodulated plasticity have been trained with evolutionary algorithms (e.g. [12, 13]; see [14] for a review). However, many of the spectacular recent advances in machine learning make use of gradient-based methods rather than evolution (which has to discover the gradients through random weight-space exploration). Thus we would like to be able to train neuromodulated plastic networks with gradient descent.

2 Methods

2.1 Background: Differentiable Hebbian plasticity

The present work builds upon the existing differentiable plasticity framework [15, 16], which allows gradient descent to optimize not just the weights, but also the plasticity of each connection. In this

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

framework, each connection in the network is augmented with a Hebbian plastic component that grows and decays automatically as a result of ongoing activity. In effect, each connection contains a fixed and a plastic component:

$$x_j(t) = \sigma\Big\{\sum_{i \in \text{inputs to } j} (w_{i,j} + \alpha_{i,j}\text{Hebb}_{i,j}(t))x_i(t-1)\Big\} \quad (1)$$

$$\text{Hebb}_{i,j}(t+1) = \text{Clip}(\text{Hebb}_{i,j}(t) + \eta x_i(t-1)x_j(t)), \quad (2)$$

where $x_i(t)$ is the output of neuron i at time t, σ is a nonlinearity (we use $\tanh$ in all experiments), $w_{i,j}$ is the baseline (non-plastic) weight of the connection between neurons i and j, and $\alpha_{i,j}$ is the *plasticity coefficient* that scales the magnitude of the plastic component of the connection. The plastic content is represented by the *Hebbian trace* $\text{Hebb}_{i,j}$, which accumulates the product of pre- and post-synaptic activity at connection i, j, as shown in Eq. 2. Note that the Clip function clips the value of $\text{Hebb}_{i,j}$ within $[-1, 1]$ (in previous work [16] we used a decay term or a normalization implementing Oja's rule, but a simple hard clip turned out to produce equal or superior performances for the tasks described here).

$\text{Hebb}_{i,j}$ is initialized to zero at the beginning of each episode/lifetime, and is updated automatically according to Eq. 2: it is a purely episodic/intra-life quantity. By contrast, $w_{i,j}$, $\alpha_{i,j}$ and η are the structural components of the network, which are optimized by gradient descent between episodes/lifetimes to minimize the expected loss over an episode[1]. Thus, in contrast to other approaches using *uniform* plasticity [17], including "fast weights" [18], the amount of plasticity in each connection (represented by $\alpha_{i,j}$) is *trainable* [16].

2.2 Backpropamine: Differentiable neuromodulation of plasticity

We propose two methods to introduce neuromodulated plasticity within the differentiable plasticity framework. In both cases, plasticity is modulated on a moment-to-moment basis by a network-controlled neuromodulatory signal $M(t)$. Here $M(t)$ is simply an output neuron of the network, though more complex choices are possible.

Method 1: Simple neuromodulation
The simplest way to introduce neuromodulation of plasticity in this framework is to make the (global) η parameter depend on the output of one or more neurons in the network. Because η essentially determines the rate of plastic change, placing it under network control allows the network to determine how plastic connections should be at any given time. Thus, Eq. 2 is replaced with

$$\text{Hebb}_{i,j}(t+1) = Clip(\text{Hebb}_{i,j}(t) + M(t)x_i(t-1)x_j(t)). \quad (3)$$

Method 2: Retroactive neuromodulation and eligibility traces
More complex schemes are possible. For example, in animal brains, dopamine was shown to retroactively gate the plasticity induced by *past* activity, within a short time window of about 1s [8, 6, 19, 20]. Thus, Hebbian plasticity does not directly modify the synaptic weights, but creates a fast-decaying "potential" weight change, to be incorporated into the actual weights only if the synapse receives dopamine within a short time window. As a result, biological Hebbian traces essentially implement a so-called *eligibility trace* [21], keeping memory of which synapses contributed to recent activity, while the dopamine signal modulates the transformation of these eligibility traces into actual plastic changes. This mechanism of reinforcement learning through reward-modulated Hebbian learning has been modelled in computational neuroscience studies, e.g. [22, 23, 24, 25, 26] (see [27] for a recent review).

Our framework easily accommodates this more refined model of dopamine effects on plasticity. We simply replace Eq. 2 above with the two equations,

$$\text{Hebb}_{i,j}(t+1) = Clip(\text{Hebb}_{i,j}(t) + M(t)E_{i,j}(t)) \quad (4)$$

$$E_{i,j}(t+1) = (1-\eta)E_{i,j}(t) + \eta x_i(t-1)x_j(t)). \quad (5)$$

[1]Note that $\alpha_{i,j}$ is a scale parameter, determining how large the Hebbian component can be, while η is an intra-life "learning rate", determining how fast it changes.

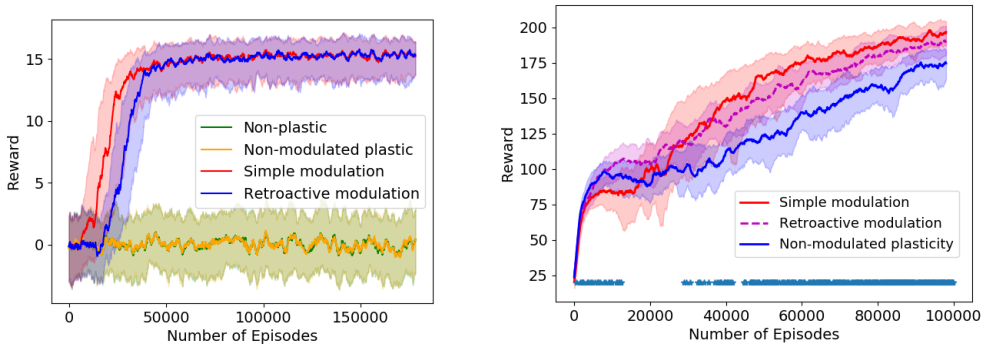

Figure 1: Left: Training curves for the cue-reward association task (medians and inter-quartile ranges of rewards per episode over 10 runs). Modulated plastic networks (red, blue) learn the task, while non-modulated and non-plastic networks (green, orange) fail. Right: Training curves for the maze exploration task: median and inter-quartile range of reward over 9 runs for each episode. Cyan stars (bottom) indicate significant difference between simple neuromodulation and non-modulated plasticity at $p < 0.05$ (Wilcoxon rank-sum test).

Here $E_{i,j}(t)$ (the eligibility trace at connection i, j) is a simple exponential average of the Hebbian product of pre- and post-synaptic activity, with trainable decay factor η. $\mathrm{Hebb}_{i,j}(t)$, the actual plastic component of the connection (see Eq. 1), simply accumulates this trace, but gated by the (signed) current value of the neuromodulatory signal $M(t)$.

3 Experiments

Task 1: Cue-reward association

Our first test task is a simple meta-learning problem that emulates an animal behavioral learning task. In each episode, one of four possible *cues* (i.e. binary 20-bit vectors randomly generated at the start of each episode) is arbitrarily chosen as the *Rewarded* cue. Repeatedly, the agent is shown two cues in succession, randomly chosen from the possible four, then a separate, fixed *Go* cue, during which the agent must respond 1 if the Rewarded cue was part of the presented pair, or 0 otherwise. A correct response produces a reward of 1.0, while an incorrect response returns reward -1.0 (a response of either 1 or 0 is always collected). This process iterates for the duration of the episode, which is 120 time steps. The architecture is a simple recurrent network with 200 neurons in the hidden recurrent layer. Only the recurrent layer is plastic: input and output weights are non-plastic, having only $w_{i,j}$ coefficients. See the Appendix for implementation details.

Training curves are shown in Figure 1 (each curve shows the median and inter-quartile range over 10 runs). Neuromodulatory approaches succeed in learning the task, while non-neuromodulatory networks (that is, the approach described in preexisting work on differentiable Hebbian plasticity [15, 16]) and non-plastic, simple recurrent networks fail to learn it. We hypothesize that this dramatic difference is related to the relatively high dimensionality of the input cues (see Appendix).

Task 2: Maze navigation task

For a more challenging problem, we also tested the approach on the grid maze exploration task introduced by [16]. Here, the maze is composed of 9×9 squares, surrounded by walls, in which every other square (in either direction) is occupied by a wall. Thus the maze contains 16 wall squares, arranged in a regular grid (see Appendix). The shape of the maze is fixed and unchanging over the whole task. At each episode, one non-wall square is randomly chosen as the reward location. When the agent hits this location, it receives a reward and is immediately transported to a *random* location in the maze. Each episode lasts 200 time steps, during which the agent must accumulate as much reward as possible. The reward location is fixed within an episode and randomized across episodes. The architecture is the same as for the previous task, but with only 100 hidden neurons in the recurrent layer and a different number of inputs and outputs (see Appendix). Results in Figure 1 (right) show that modulatory approaches again outperform non-modulated plasticity.

Table 1: Test Perplexity Results on Penn-Tree Bank (mean and 95% CI over 16 runs). Lower values are better; each model has the same number of total parameters. Retroactive neuromodulation outperforms basline LSTMs by 1.7 perplexity point ($p < 1e-7$, Wilcoxon rank-sum test)

Model	Test Perplexity
Baseline LSTM [29]	104.26 ± 0.22
LSTM with Differential Plasticity	103.80 ± 0.25
LSTM with Simple Neuromodulation	102.65 ± 0.30
LSTM with Retroactive Neuromodulation	102.48 ± 0.28

Task 3: Language Modeling

Word-level language modeling is a challenging supervised learning sequence problem, where the goal is to predict the next word in a large language corpus. Language modeling requires storing long term context, and therefore LSTM models [28] generally perform well on this task [29]. The goal of this experiment is to study the benefits of adding plasticity and neuromodulation to LSTMs.

We used the Penn Tree Bank corpus (PTB), a well known benchmark for language modeling [30] consisting of 929k training words, 73k validation words, and 82k test words, with a vocabulary of 10k words. Each network consists of an embedding layer, followed by two LSTM layers, comprising approximately 200 neurons each; exact size is adjusted to ensure that the total number of trainable parameters (4.8 million) remains constant across all experiments. The final layer is a softmax layer of size 10k. Training occurs with backpropagation through time over 20 time steps unrollings (see Appendix for details).

We tested four different models (Table 1). (1) *Baseline LSTM* model, as described in the previous paragraph (similar to [29]. (2) *LSTM with Differentiable Plasticity*: we add plasticity (Eqs. 1, 2) to the recurrent connections of the LSTM, i.e. the h-to-c connections. Each connection has its own trainable $\eta_{i,j}$ (see Appendix for details). (3) *LSTM with Simple Neuromodulation:* we add simple neuromodulation (Eq. 3) the plastic h-to$c$ connections. The η parameters are replaced by an output neuron $M(t)$ (one separate $M(t)$ for each LSTM layer). (4) *LSTM with Retroactive Neuromodulation*: we use the retroactive modulation described in Eqs. 4 and 5. See Appendix for details.

For each of the four models, we separately searched for the best hyperparameters with equally-powered grid-search. Each model was then run 16 times with its best hyperparameter settings. Results (Table 1) show that plastic LSTMs slightly, but significantly outperform Baseline LSTMs (Wilcoxon rank-sum test, $p = 0.0044$). Retroactively neuromodulated LSTMs outperform non-modulated plastic LSTMs ($p = 1e-6$), and provides about 1.7 perplexity improvement vs. the Baseline LSTM ($p = 1e-7$). Retroactive neuromodulation slightly outperforms simple neuromodulation, but failed to reach significance ($p = 0.066$).

4 Discussion and Future Work

In this paper, for the first time, we show how neuromodulated plastic networks can be trained with gradient descent, opening up a new research direction into optimizing large-scale self-modifying neural networks. A finding of note is that plastic and neuromodulated LSTMs outperform standard LSTMs on a benchmark language processing task. Importantly, such tasks are a central domain of application of LSTMs with considerable potential economic impact.

Conceptually, an important reference is the "Learning to Reinforcement Learn" (L2RL) framework of [31, 32], in which a network is meta-trained over many episodes (with A2C) to store relevant information within its hidden activity state during each episode/lifetime: no weight changes occur after training. First, our approach adds flexibility to this model, by allowing state information to be stored with weight changes in addition to hidden state changes. But besides this benefit, our framework (which allows the network to update its own connectivity) might potentially replace the outer A2C loop with as a learned, arbitrary reward-based weight-modification scheme, which might make use of any signal the network can compute – reward predictions, unexpectedness, saliency, etc.

In conclusion, with a rich potential for applications and extensions, our framework for neuromodulated plastic networks opens many avenues of exciting research.

References

[1] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-To-End memory networks. In C Cortes, N D Lawrence, D D Lee, M Sugiyama, and R Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc., 2015.

[2] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *NIPS Workshop on Meta-Learning*, 2017.

[3] Stephen J Martin, Paul D Grimwood, and Richard GM Morris. Synaptic plasticity and memory: an evaluation of the hypothesis. *Annual review of neuroscience*, 23(1):649–711, 2000.

[4] Xu Liu, Steve Ramirez, Petti T Pang, Corey B Puryear, Arvind Govindarajan, Karl Deisseroth, and Susumu Tonegawa. Optogenetic stimulation of a hippocampal engram activates fear memory recall. *Nature*, 484(7394):381, 2012.

[5] Paolo Calabresi, Barbara Picconi, Alessandro Tozzi, and Massimiliano Di Filippo. Dopamine-mediated regulation of corticostriatal synaptic plasticity. *Trends Neurosci.*, 30(5):211–219, May 2007.

[6] Kaiwen He, Marco Huertas, Su Z Hong, Xiaoxiu Tie, Johannes W Hell, Harel Shouval, and Alfredo Kirkwood. Distinct eligibility traces for LTP and LTD in cortical synapses. *Neuron*, 88(3):528–538, November 2015.

[7] Shaomin Li, William K Cullen, Roger Anwyl, and Michael J Rowan. Dopamine-dependent facilitation of LTP induction in hippocampal CA1 by exposure to spatial novelty. *Nature Neuroscience*, 6(5):526–531, May 2003.

[8] S Yagishita, A Hayashi-Takagi, G C R Ellis-Davies, H Urakubo, S Ishii, and H Kasai. A critical time window for dopamine actions on the structural plasticity of dendritic spines. *Science*, 345(6204):1616–1620, September 2014.

[9] Katiuska Molina-Luna, Ana Pekanovic, Sebastian Röhrich, Benjamin Hertler, Maximilian Schubring-Giese, Mengia-Seraina Rioult-Pedotti, and Andreas R Luft. Dopamine in motor cortex is necessary for skill learning and synaptic plasticity. *PLoS One*, 4(9):e7082, September 2009.

[10] S L Smith-Roe and A E Kelley. Coincident activation of NMDA and dopamine D1 receptors within the nucleus accumbens core is required for appetitive instrumental learning. *J. Neurosci.*, 20(20):7737–7742, October 2000.

[11] Anatol C Kreitzer and Robert C Malenka. Striatal plasticity and basal ganglia circuit function. *Neuron*, 60(4):543–554, November 2008.

[12] Andrea Soltoggio, John A Bullinaria, Claudio Mattiussi, Peter Dürr, and Dario Floreano. Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In *Proceedings of the 11th international conference on artificial life (Alife XI)*, number LIS-CONF-2008-012, pages 569–576. MIT Press, 2008.

[13] Sebastian Risi and Kenneth O Stanley. A unified approach to evolving plasticity and neural geometry. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–8. IEEE, 2012.

[14] Andrea Soltoggio, Kenneth O. Stanley, and Sebastian Risi. Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. 2017.

[15] T. Miconi. Backpropagation of hebbian plasticity for continual learning. In *NIPS Workshop on Continual Learning*, 2016.

[16] Thomas Miconi, Jeff Clune, and Kenneth O. Stanley. Differentiable plasticity: training plastic networks with gradient descent. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[17] J. Schmidhuber. Reducing the ratio between learning complexity and number of time varying variables in fully recurrent nets. In Stan Gielen and Bert Kappen, editors, *ICANN '93: Proceedings of the International Conference on Artificial Neural Networks Amsterdam, The Netherlands 13–16 September 1993*, pages 460–463. Springer London, London, 1993.

[18] Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and

R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4331–4339. 2016.

[19] Simon D Fisher, Paul B Robertson, Melony J Black, Peter Redgrave, Mark A Sagar, Wickliffe C Abraham, and John N J Reynolds. Reinforcement determines the timing dependence of corticostriatal synaptic plasticity in vivo. *Nature Communications*, 8(1):334, August 2017.

[20] Stijn Cassenaer and Gilles Laurent. Conditional modulation of spike-timing-dependent plasticity for olfactory learning. *Nature*, 482(7383):47–52, January 2012.

[21] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

[22] Eugene M Izhikevich. Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cereb. Cortex*, 17(10):2443–2452, October 2007.

[23] Gregor M Hoerzer, Robert Legenstein, and Wolfgang Maass. Emergence of complex computational structures from chaotic neural networks through reward-modulated hebbian learning. *Cereb. Cortex*, 24(3):677–690, March 2014.

[24] Ila R Fiete, Michale S Fee, and H Sebastian Seung. Model of birdsong learning based on gradient estimation by dynamic perturbation of neural conductances. *J. Neurophysiol.*, 98(4):2038–2057, October 2007.

[25] Andrea Soltoggio and Jochen J Steil. Solving the distal reward problem with rare correlations. *Neural Comput.*, 25(4):940–978, April 2013.

[26] Thomas Miconi. Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks. *Elife*, 6, February 2017.

[27] Wulfram Gerstner, Marco Lehmann, Vasiliki Liakoni, Dane Corneil, and Johanni Brea. Eligibility traces and plasticity on behavioral time scales: Experimental support of NeoHebbian Three-Factor learning rules. *Front. Neural Circuits*, 12:53, July 2018.

[28] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory, 1997.

[29] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014.

[30] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, June 1993.

[31] Jane X. Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Rémi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. 2016.

[32] Jane X Wang, Zeb Kurth-Nelson, Dharshan Kumaran, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Demis Hassabis, and Matthew Botvinick. Prefrontal cortex as a meta-reinforcement learning system. *Nature neuroscience*, 21(6):860, 2018.

[33] Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations*, 2018.

[34] Y. Gal. A theoretically grounded application of dropout in recurrent neural networks. 2015.

[35] Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. 2016.

A Appendix

A.1 Language modeling experiments and plastic LSTMs

A.1.1 Adding plasticity to LSTM

Each LSTM node consists of four weighted recurrent paths through i_t, j_t, f_t and o_t as shown in the equations below:

$$i_t = \tanh(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{6}$$

$$j_t = \sigma(W_{xj}x_t + W_{hj}h_{t-1} + b_j) \tag{7}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{8}$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{9}$$
$$c_t = f_t \otimes c_{t-1} + i_t \otimes j_t \tag{10}$$
$$h_t = \tanh(c_t) \otimes o_t \tag{11}$$

j_t, f_t and o_t are used for controlling the data-flow through the LSTM and i_t is the actual data. Therefore, plasticity is introduced in the path that goes through i_t (adding plasticity to the control paths of LSTM is for future-work) . The corresponding pre-synaptic and post-synaptic activations (denoted by $x_i(t-1)$ and $x_j(t)$ respectively in equations 1 and 2) are h_{t-1} and i_t. A layer of size 200 has 40k (200×200) plastic connections. Each plastic connection has its own individual η (used in equation 2) that is learned through backpropagation. The plasticity coefficients ($\alpha_{i,j}$) are used as shown in equation 1.

A.1.2 Adding neuromodulation to LSTM

As shown in equation 3, for simple neuromodulation, the η is replaced by the output of a network computed neuron $M(t)$. For neuromodulated LSTMs, individual η for each plastic connection is replaced by the output of a neuron ($M(t)$) that has a fan-out equal to the number of plastic connections. The input to this neuron is the activations h_{t-1} of the layer from the previous time-step. Each LSTM layer has its dedicated neuromodulatory neuron. Other variations of this setting include having one dedicated neuromodulatory neuron per node or having one neuromodulatory neuron for the whole network. Preliminary experiments showed that these variations performed worse and therefore they were not further evaluated.

A.1.3 More details for Language Modeling experiment

All the four models presented in Table 1 are trained using SGD. Initial learning rate was set 1.0. Each model is trained for 13 epochs. The hidden states of LSTM are initialized to zero; the final hidden states of the current minibatch are used as the initial hidden states of the subsequent minibatch.

Grid-search was performed for four hyperparameters: (1) Learning rate decay factor in the range 0.25 to 0.4 in steps of 0.01. (2) Epoch at which learning rate decay begins in the range - $\{4, 5, 6\}$. (3) Initial scale of weights in the range - $\{0.09, 0.1, 0.11, 0.12\}$. (4) L2 penalty constant in the range - $\{1e-2, 1e-3, 1e-4, 1e-5, 1e-6\}$.

The state-of-art results in language modeling domain have been achieved with much larger LSTM models [33]. In such large models, recurrent dropouts [34] need to be introduced for regularization. An interesting direction for future-work is to understand the effects of dropout on plastic and neuromodulated connections.

A.2 Cue-reward association task

A.2.1 Task description

Here we provide a more complete description of the cue-reward association task described in section 3. At the beginning of each episode, four 20-bit binary vectors are randomly generated; these are the *cues* for this episode. One of these four cues is randomly chosen as the *Rewarded* cue for the episode. Then, during the episode, repeatedly, two randomly chosen cues are presented to the network in succession, followed by a *Go* cue (which is fixed across episodes and different from all other input cues). After the *Go* cue is shown, the network's response is collected (a response is always chosen, i.e. this is a two-alternative forced choice task). This response must be 1 ("press lever") if the *Rewarded* cue was part of the previously presented pair, and 0 ("don't press lever") otherwise. I this is the case, a positive reward of +1 is delivered to the network; otherwise, a negative response of -1 is delivered instead (this reward is perceived by the agent at the next time step, following common meta-learning practice [31, 35]) This process iterates for the duration of the episode, which is 120 time steps.

To prevent simple time-locked scheduling strategies, a variable number of zero-input time steps are inserted, including at least two consecutive ones after each presentation of the Go cue, plus a uniformly chosen number of zero-input time steps between zero and three inclusive, inserted at random times between two Go cue presentations. As a result, the length of each trial varies, and the number of trials per episode is somewhat variable (the mean number of trials per episode is ∼18).

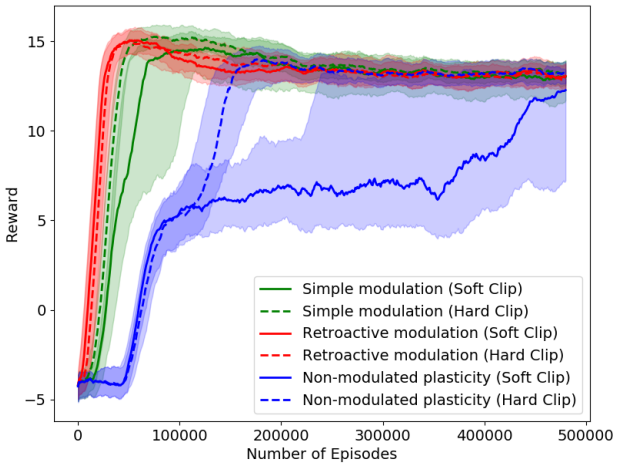

Figure 2: Training curves for the cue-reward association task with fixed, binary four-bit cues (medians and inter-quartile ranges of rewards per episode over 10 runs). "Soft clip" refers to a different clipping operation used in Equation 2; "Hard clip" is the same as used in the present paper, i.e. the simple clipping described in Methods. Note that non-modulated plastic netowkr succeed in solving this task.

The architecture is a simple recurrent network with 200 neurons in the hidden recurrent layer. Only the recurrent layer is plastic: input and output weights are non-plastic, having only $w_{i,j}$ coefficients. There are 24 inputs: 20 binary inputs for the current cue and one input providing the time elapsed since the start of the episode, as well as two binary inputs for the one-hot encoded response at the previous time step and one real-valued channel for the reward received at the previous time step, in accordance with common meta-learning practice [31, 35]. There are four outputs: two binary outputs for the one-hot encoded response, plus an output neuron that predicts the sum of future discounted rewards $V(t)$ over the remainder of the episode (as mandated by the A2C algorithm that we use for meta-training, following [31]), and the neuromodulatory signal $M(t)$. The two response outputs undergo a softmax operation to produce probabilities over the response, while the $M(t)$ signal is passed through a $\tanh$ nonlinearity and the $V(t)$ output is a pure linear output. All gradients are clipped at norm 2.0, which greatly improved stability.

A.2.2 Hypothesis about the failure of non-modulated plastic networks

In this task, neuromodulated plasticity was able to learn a task that non-modulated plasticity simply could not. What might be the source of this difference? In a previous experiment, we implemented the same task, but using only four fixed 4-bit binary cues for the entire task, namely, '1000', '0100', '0010' and '0001'. In this simplified version of the task, there is no need to memorize the cues for each episode, and the only thing to be learned for each episode is which of the four known cues is associated with reward. This is in contrast with the version used in the paper above, in which the cues are arbitrary 20-bits vectors randomly generated for each episode. With the fixed, four-bit cues, non-modulated plasticity was able to learn the task, though somewhat more slowly than neuromodulated plasticity (see Figure 2).

This suggests neuromodulated plasticity could have a stronger advantage over non-modulated plasticity specifically in situations where the association to be learned involves arbitrary high-dimensional cues, which must be memorized jointly with the association itself. This echoes the results of [16], who suggest that plastic networks outperform non-plastic ones specifically on tasks requiring the fast memorization of high-dimensional inputs (e.g. image memorization and reconstruction task in [16]).

Clearly, more work is needed to investigate which problems benefit most from neuromodulated plasticity, over non-modulated or non-plastic approaches. We intend to pursue this line of research in future work.

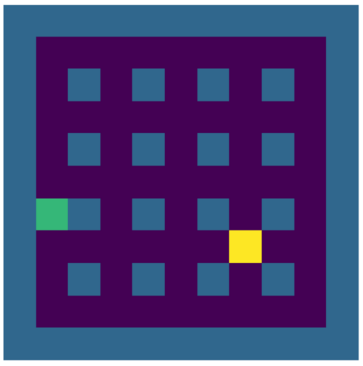

Figure 3: Layout of the maze, including an example agent location (yellow) and reward location (green, for illustration only: the reward is not visible to the agent).

A.3 Maze exploration task

Here we provide a more complete description of the grid maze exploration task discussed in section 3. Note that this task was introduced and described in [16].

Here, the maze is composed of 9×9 squares, surrounded by walls, in which every other square (in either direction) is occupied by a wall. Thus the maze contains 16 wall squares, arranged in a regular grid (see Figure 3). The shape of the maze is fixed and unchanging over the whole task. At each episode, one non-wall square is randomly chosen as the reward location. When the agent hits this location, it receives a reward and is immediately transported to a *random* location in the maze. Each episode lasts 200 time steps, during which the agent must accumulate as much reward as possible. The reward location is fixed within an episode and randomized across episodes. Note that the reward is invisible to the agent, and thus the agent only knows it has hit the reward location by the activation of the reward input at the next step (and possibly by the teleportation, if it can detect it).

The architecture is a simple recurrent neural network, similar to thge one for the cue-reward association task, but with only 100 recurrent neurons. The outputs consist of 4 action channels (i.e. one for each of the possible actions: left, right, up or down) passed through a softmax, as well as the pure linear $V(t)$ output (which predicts the expected sum of discounted rewards until the end of the episode, as required by the A2C algorithm that we use for meta-training [31]) and the $M(t)$ neuromodulatory signal passed through a $\tanh$ nonlinearity. Inputs to the agent consist of a binary vector describing the 3×3 neighborhood centered on the agent (each element being set to 1 or 0 if the corresponding square is or is not a wall), plus four additional inputs for the one-hot encoded action taken at the previous time step, and one input for the reward received at the previous time step, following common meta-learning practice [31]. Only recurrent weights are plastic: input-to-recurrent and recurrent-to-output weights are non-plastic.